sensors

MDPI

Article

Real-Time Efficient Relocation Algorithm Based on Depth Map for Small-Range Textureless 3D Scanning

Fengbo Zhu [1,*], Shunyi Zheng [1,*], Xiaonan Wang [2], Yuan He [1], Li Gui [3] and Liangxiong Gong [4]

1 School of Remote Sensing and Information Engineering, Wuhan University, Wuhan 430079, China
2 Wuhan Zhongguan Automation Technology Co., Ltd., Wuhan 430066, China
3 Collaborative Innovation Center of Geospatial Technology, Wuhan University, Wuhan 430079, China
4 Nanchang Institute of Surveying and Mapping, Nanchang 330013, China
* Correspondence: zhu_fb@whu.edu.cn (F.Z.); syzheng@whu.edu.cn (S.Z.)

Received: 10 August 2019; Accepted: 4 September 2019; Published: 6 September 2019

Abstract: As an important part of industrial 3D scanning, a relocation algorithm is used to restore the position and the pose of a 3D scanner or to perform closed-loop detection. The real time and the relocation correct ratio are prominent and difficult points in 3D scanning relocation research. By utilizing the depth map information captured by a binocular vision 3D scanner, we developed an efficient and real-time relocation algorithm to estimate the current position and pose of the sensor real-time and high-correct-rate relocation algorithm for small-range 3D texture less scanning. This algorithm mainly involves feature calculation, feature database construction and query, feature matching verification, and rigid transformation calculation; through the four parts, the initial position and pose of the sensors in the global coordinate system is obtained. In the experiments, the efficiency and the correct-rate of the proposed relocation algorithm were elaborately verified by offline and online experiments on four objects of different sizes, and a smooth and a rough surface. With more data frames and feature points, the relocation could be maintained real time within 200 ms, and a high correct rate of more than 90% could be realized. The experimental results showed that the proposed algorithm could achieve a real-time and high-correct-ratio relocation.

Keywords: real-time relocation; global relocation; key point detector; feature matching verification; transformation estimation

1. Introduction

Recently, some high-precision three-dimensional (3D) handheld scanners, such as GoScan [1], EinScan-Pro+ [2], and GSCAN [3] have been used in 3D modeling, industrial inspection, and reverse design. These 3D scanners are based on the binocular stereo visual technology, which captures the two images with structured light specular once and generates the depth map by using a stereo matching method. These devices can generate a more accurate 3D depth map because of the smaller scanning distance. However, because of the small scanning distance, the range of the depth map is small. When the scanner moves too fast, or the scanner moves directly to another location to scan the rest of the region, this small-range scanner unavoidably loses its position and pose. Restoring the pose and the position of the scanner requires the ability of relocation. The information that we can use is only the images captured by the cameras and the depth map information obtained through stereo matching. However, in the case of industrial mold scanning or gypsum sculpture scanning, the texture of the scanning object is often scarce; in other words, it is difficult to locate the scanner by using texture information. Therefore, the depth map is used to locate the scanner.

The process of scanning can be divided into tracking and mapping. In the tracking part, the current depth map is used to register with the previous frame's depth map or the depth map generated by the

raycasting method [4]. The registration result is a 6-degree of freedom (DOF) transformation matrix. In the mapping part, the global frame data are generated by transforming the current depth map by using the transformation matrix; these global frame data are then fused into the scanning model by using the truncated signed distance function (TSDF) method [4]. When the tracking is lost, the current depth map is used to relocate the initial position and pose of the scanner; then, the refined position and pose can be obtained by registering the current depth map to the model map generated by using the initial position and pose. We focused on the recovery of the global pose and position of the scanner by using a depth map when tracking failed.

Depth maps have been used for 3D object recognition [5], 3D registration [6], 3D modeling [7], 3D object search [8], etc. However, no related studies can be found on small-range sensor relocation using depth maps in the incremental scanning process. In 3D object recognition [9], an intrinsic shape signatures (ISS) feature description is proposed and a query database is built for searching similar objects, which achieves a high recognition performance; however, the focus is on finding a model similar to the current radar scan model from the computer aided design (CAD) model library, instead of getting the position and the pose. 3D pairwise registration often uses the iterative closest point (ICP) [10] or its variants [11–14] for registration to obtain the transformation relationship; however, the ICP or its variants are sensitive to the initial values. Some better rough registration methods based on features have been proposed to solve the initial value problem [15–17]; these algorithm pipelines can generally be summarized by the following three steps: (1) Point cloud features such as globally aligned spatial distribution method (GASD) [18], ISS, and signature of histograms of orientations (Shot) [19] are extracted from the point clouds. (2) The initial matching features are obtained, which meet the similar distance measure of the feature descriptor. (3) Some matching verification methods, such as Hough vote [15], geometric consistency (GC) [20], and search of inliers (SI) [21], are used to filter out the wrong matching features and obtain the registration matrix. Although these feature-based coarse registration methods solve the initial value problems, they are not suitable for incremental 3D reconstruction, because as the amount of scan data increases, the model becomes larger, and thus extracting the feature points and finding the initial matching points on the model will be a very time-consuming process.

Features are successfully applied in the coarse registration method. They are divided into global features and local features [22]. Global features are not suitable for point cloud pair registration, which only has a partially overlapping region [22]. The local features describe the spatial or geometric properties of the neighborhood of the feature point. Compared with the global features, the local features are more suitable for pairwise alignment [22]. Therefore, these features were used to recover the position and the pose of the sensors in this study. Then, we were devoted to solve the following four problems: (1) How to extract the feature points from the current frame depth map effectively? (2) How to construct the feature query database and query the features quickly? (3) How to filter out the wrong matching feature point pair effectively? (4) How to get a correct transformation matrix?

A few local feature extreme points that represent the geometric characteristics of the whole data are extracted, which can reduce the information redundancy. Thus far, 3D feature extraction has been extensively studied, and many different 3D feature descriptors [23–27] have been proposed. The local reference frame (LRF) is an independent coordinate system constructed on the local 3D surface. It is broadly used in 3D local feature descriptors, and the LRF can make the local feature descriptor with rotational invariance. Many different LRF calculation methods [28–33] have also been proposed to get a rotation-invariant local feature descriptor. The stability of the extracted feature points is an important problem. The eigenvalues ratio and the single eigenvalue constraint are used to improve the stability of the feature points [9,34,35]. In the actual scene scan, the depth map of each frame contains some invalid data because of occlusion, deep slot, and sharp protuberance. The eigenvalues constraint and the eigenvalues ratio constraint usually lead to the location of the feature points on the edge of the invalid data. The scanning of different perspectives leads to different invalid data edges, which will result in unstable features. These unstable features affect the feature matching and transformation calculations.

An efficient similar feature search strategy is also an important issue in the incremental reconstruction process. An appropriate feature organization structure can accelerate the similar feature queries and insertions. The easiest way to do so is to perform a brute-force search to query all the similar features, but as the number of feature points increases, the search time will increase rapidly. The bags of binary words (DBoW2) [36] method is commonly used in relocation and closure detection in simultaneous localization and mapping (SLAM). However, as an offline training method, it cannot be applied to different unknown scenes. KD-tree [37] is a fast and accurate method to get the nearest feature; however, KD-tree needs to be rebuilt every time when new feature points are added to the tree, which costs time. The local sensitive hash (LSH) [38–40] method is a method of approximate nearest neighbor (ANN), which is mainly applied for offline image retrieval. A local sensitive hash tree (LST) [9] is used to search for similar objects; although this method is an ANN method, its structure is very beneficial for feature query and insertion.

In the scanning process, similar features may be distributed in different spatial locations; this makes it difficult to query the correct matching of the feature. Some matching verification methods should be used to get the correct matching feature from a large number of similar features. Some methods, such as the nearest neighbor (NN) and the nearest neighbor similarity ratio (NNSR) [41], use the feature description distance measure to get the matching feature, but they ignore the geometric consistency. The GC algorithm uses feature distance verification to get the similar features and uses clustering methods to verify the geometric consistency. A 3D Hough vote [15] is a good method to verify the geometric consistency; however, it needs to allocate a large amount of memory when the target object is large. The random sampling method [42] can obtain the model fitting result with the greatest confidence, but when there are a large number of mismatched point pairs, it may get wrong results. The SI method uses a feature descriptor distance ratio limitation, point pair distance constraints, and a global voting method of point-to-point transformation to construct a global vote, and the adaptive thresholding method is used to obtain the inner points. The consistency ranking vote method [43] ranks the scores of the similar features, and the geometric consistency measure is used to get the correct matching features. We proposed a new verification method: first, we used the NN method to get the similar features, and then, the mismatched features were filtered out by the distance, angle, and direction verification. Finally, the cluster method was used to get the clustered group with a high confidence.

The 6-DOF transform optimization is a mature method. When the matching feature point pairs with high confidence are obtained, the optimized transformation matrix can be quickly obtained by the ICP or its variant methods combined with the RANSAC method. However, when there are duplicate structures in the scene, the random sampling method produces different and reliable results, which may cause a relocation failure.

We developed a real-time and efficient relocation pipeline. In the feature calculation part, more constraints were used to avoid the feature points from falling in an unstable place. Then, LST was used to perform the similar feature query and insertion. After the similar features were queried by LST, a distance, angle, and direction verification method was used to filter out some mismatching feature point pairs, and the cluster method was used to get the clustered group with a high confidence. Finally, we used the improved RANSAC sample method to avoid the interference of the duplicate structures. Therefore, this paper makes the following contributions:

- This paper presents a real-time and high-correct-rate algorithm pipeline for relocation by using depth maps in the incremental reconstruction scanning process. We divided the relocation algorithm into four parts for the first time: feature calculation, feature database construction and query, corresponding feature verification, and rigid transformation calculation.
- In the feature calculation step, we used the normal to get the LRF, and the feature descriptor was normalized to reduce the effect of the different resolutions and scanning distances. The area limitation was used to get a more stable feature point. In the feature matching verification part, we added an eigenvalue limitation to quickly get a coarse matching feature. The distance, angle, and direction verification method and the dynamic cluster method were used to filter the error

matching feature pairs. In the rigid transformation calculation part, we improved the RANSAC method to avoid the interference of similar structures.

The remainder of this paper is organized as follows: the details of the relocation pipeline are presented in Section 2. In Section 3, we provide relocation experiments and discussions. Finally, we draw conclusions and outline some ideas for future work in Section 4.

2. Methods

After reviewing the related work on feature calculation, feature organization and query, feature matching verification, and transformation calculation, we focused our work on the real-time and high-correct-ratio global relocation algorithm.

The pipeline of the proposed relocation algorithm is shown in Figure 1. As shown, the current depth map captured from the scanner was downsampled to accelerate the computation. Then, features were extracted from the downsampling depth map. When the current frame tracking succeeded, the features were added directly to the feature database. When the current frame tracking failed, the current frame features were used to query similar features from the feature database, the match verification part filtered some incorrect matches, and the rest of the corresponding feature point pairs were used to calculate the transformation matrix that transformed the current frame to the global model coarsely. The transformation matrix could be viewed as the initial global position and pose of the scanner. The ordered details on the relocation algorithm of the proposed pipeline are as follows.

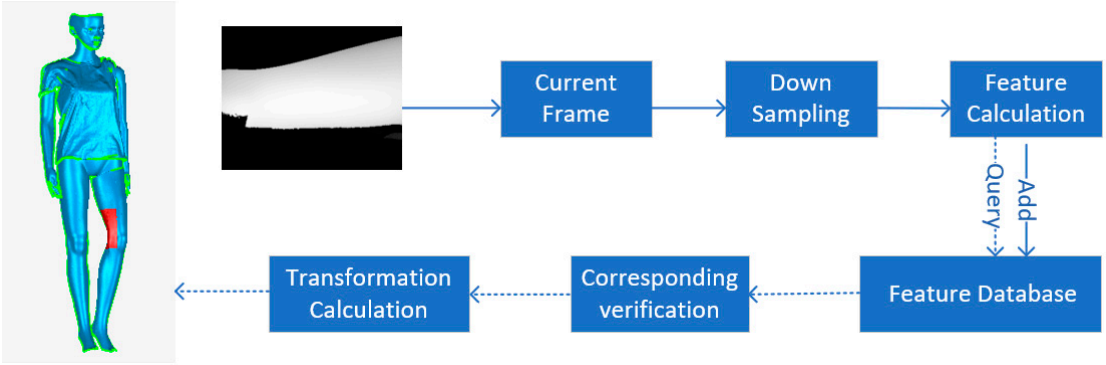


Figure 1. Proposed global relocation algorithm pipeline.

2.1. Downsampling

The downsampling method was used to reduce the data and accelerate the calculation, which can be expressed as follows:

$$\begin{cases} w' = w/s \\ h' = h/s \\ focus' = focus/s \\ x_0' = x_0/s \\ y_0' = y_0/s \\ depthM' = submap(depthM) \end{cases} \quad (1)$$

where s is the sample rate; $submap()$ function represents that the depth map is downsampled at equal intervals of s. Further, w, h, $focus$, (x_0, y_0), and $depthM$ represent the width, the height, the focus, the image principal point, and the depth map data, respectively. Next, w', h', $focus'$, $\left(x_0', y_0'\right)$, and $depthM'$ represent the width, height, focus, image principal point, and depth map data, respectively, after the depth map downsampling. Through experimental observations, we found that the setting of $s = 4$ still maintained a satisfactory performance.

2.2. Feature Calculation

An LRF is a 3D coordinate system established on the local surface. It is important because feature descriptions based on the LRF are invariant to a rigid transformation [44]. The neighboring points of point p were used to calculate LRF by the covariance decomposition (CA). In this paper, the LRF was calculated on the basis of the covariance decomposition of the normals of the neighbor points. The advantage was obvious: the sum of three eigenvalues of CA was 1, which reduced the difficulty of setting the thresholds. The method of CA was used to calculate the normal direction of each point on the depth map as follows:

$$Cov(p) = \frac{1}{N(p)} \sum_{q \in N(p)} (q - \overline{q})(q - \overline{q})^T \tag{2}$$

where q is the point in the neighborhood of point p, $N(p)$ is the number of neighboring points, and $\overline{q}$ is the average position of the neighboring points. The eigenvectors and eigenvalues could be calculated from CA of $Cov(p)$; the normal n_q is the eigenvector corresponding to the minimum eigenvalue. Moreover, the normal keeps the direction towards the viewpoint. Then, the normals of the neighboring points were used to obtain LRF as follows:

$$Cov(n) = \frac{1}{N(p)} \sum_{q \in N(p)} n_q {n_q}^T \tag{3}$$

The eigenvalues $\lambda_1, \lambda_2,$ and λ_3 ($\lambda_1 > \lambda_2 > \lambda_3$) were obtained from the CA of $Cov(n)$, the eigenvector $lrf_x = \{x', y', z'\}$ corresponding to the eigenvalue λ_3 was taken as the local x-axis direction of the feature, and the feature vector $lrf_y = \{x'', y'', z''\}$ corresponding to the feature value λ_2. was used as the feature's local y-axis direction. The eigenvector $lrf_z = \{x''', y''', z'''\}$ corresponding to the eigenvalue λ_1 was taken as the local z-axis direction of the feature.

2.2.1. LRF Disambiguation

Firstly, we removed the ambiguity of the z-axis of the LRF. The direction of z-axis could be corrected by the position of the viewpoint. In each depth map, the center point of the cameras was viewed as the viewpoint, in the coordinate system of the camera. When the position of the camera center was (0, 0, 0), the direction of the z-axis was determined as follows:

$$lrf_z = \begin{cases} \{x''', y''', z'''\} z > 0 \\ \{-x''', -y''', -z'''\} z < 0 \end{cases} \tag{4}$$

Secondly, we removed the ambiguity of the x-axis of the LRF. We obtained the x-axis direction by using this method described in the triple orthogonal local depth images (TOLDI) [33]. Then, the y-axis direction was determined by using the cross-product of z-axis and x-axis.

2.2.2. Feature Extreme Point Extraction

Compared with the feature extreme point selection method based on the neighborhood [15], the grid-based selection method was faster. The current frame was divided into N × M grids, and a feature extreme point was expected to be selected in each grid. The feature extreme points had to satisfy the following conditions:

1. The ratio of the number of neighborhood points to the size of the neighborhood window should meet the threshold. Assuming that the neighborhood radius is r and the depth of the current point is z, the r/focus × z depth represents the radius of the pixel window. Further, assuming that the width and the height of the window are w and h and the number of points in the radius r is num, the ratio should satisfy $num/(W \times H) > \tau_{area}$.

2. In each grid, the feature points with the maximum λ_3 are selected as the local feature extreme points. In order to avoid the interference of the feature extreme points falling on the edge of the grid, the feature extreme point should be verified in its eight neighborhoods.
3. The feature extreme point should satisfy the ratio threshold $\lambda_1/\lambda_2 > \tau_{\lambda_{12}}, \lambda_2/\lambda_3 > \tau_{\lambda_{23}}$.
4. The feature extreme point should satisfy the single-eigenvalue constraints $\lambda_2 > \tau_{\lambda_2}, \lambda_3 > \tau_{\lambda_3}$.

Condition 1 can effectively exclude the feature points with holes in the neighborhood or those falling on the edge. Condition 3 can exclude the ambiguity due to symmetry. Condition 4 can ensure that the feature description has the discrimination ability.

2.2.3. Feature Description

In this study, we partially modified the feature descriptor mentioned in ISS [9]. ISS counts the number of neighboring points of the feature point and uses the reciprocal of the number as the weight. This method is time-consuming. We used the distance as the weight. In the actual scanning process, spatial resolution is determined by the distance from the sensor to the object; therefore, in this study, the features were normalized to reduce the effect of the resolutions.

ISS used a discrete spherical grid recursively computed from a base octahedron to divide the spherical angular space into relatively uniformly and homogeneously distributed cells and the radial distances were also evenly divided [9]. The neighborhood points were transformed into the partition space by the LRF, each cell of the ISS descriptor was constructed by counting the weight of the falling point. The weight formula we used is as follows:

$$\mathrm{w} = \mathrm{r} - \|d_i\| \tag{5}$$

where r represents the query radius of the feature point neighborhood and d_i indicates the distance between the neighborhood point and the feature point. After calculating the cumulative weight in each bin, the feature descriptor was normalized; this normalization reduced the effect of the different resolutions caused by the scanning distances.

2.2.4. Feature De-Duplication

Feature de-duplication helps to reduce the number of features and to speed up query. Firstly, we used the hash map method to reduce de-duplication. The virtual space was divided into voxels, and the global position of the feature point was mapped to a hash bin according to Formula (6). If the hash bin had already been identified, this feature point would be removed as redundancy; if not, this hash bin would be identified. Assuming that the edge length of the divided virtual voxel was u, we mapped the feature point $p(x, y, z)$ in the virtual space as follows:

$$\mathrm{Hashpos} = \mathrm{hash}(\lfloor x/u \rfloor, \lfloor y/u \rfloor, \lfloor z/u \rfloor) \tag{6}$$

After reducing the feature redundancy, there were still some feature points that were close to each other. The non-maximum suppression method was used to further reduce the number of redundant feature points. In the neighborhood of each feature point, the feature point with the largest λ_3 value was selected as the local feature extreme point, and the other feature points were removed as redundant points.

2.3. Feature Database Construction and Query

The construction of the feature query tree directly adopted the LST method proposed in the ISS [9], because this algorithm can quickly organize and query the feature points.

2.4. Corresponding Feature Verification

2.4.1. Feature Eigenvalue and Descriptor Verifications

The LST cannot guarantee that the features queried from LST are all correct matches for the current query features, so some methods are needed to filter out the incorrect matching features. As the covariance calculated by the normal of the neighborhood points was used, the sum of the three eigenvalues of the feature point was 1; this made the threshold easier to set. Assume that the descriptor of feature p was f_p, the eigenvalue of feature point p was λ_1^p, λ_2^p, λ_3^p, the corresponding query point was $\left\{p'_i, \ldots, p'_j\right\}$, and the descriptor of feature p'_k was f'_k. The eigenvalues of feature point p'_k were $\lambda_1^{p'_k}$, $\lambda_2^{p'_k}$, $\lambda_3^{p'_k}$. Assume that the eigenvalue threshold was ν_{λ_1},ν_{λ_2},ν_{λ_3}. Then, eigenvalues were used to filter out some dissimilar features as follows:

$$\begin{cases} \lambda_1^p - \lambda_1^{p'_k} < \nu_{\lambda_1} \\ \lambda_2^p - \lambda_2^{p'_k} < \nu_{\lambda_2} \\ \lambda_3^p - \lambda_3^{p'_k} < \nu_{\lambda_3} \end{cases} \tag{7}$$

The feature descriptor was normalized, and if two features were the same, the score was 1. We used the following formula to filter out the dissimilar features.

$$\begin{cases} s\left(p, p'_k\right) = f * f'_k \\ s\left(p, p'_k\right) > \tau_s \end{cases} \tag{8}$$

where τ_s is the threshold of the similarity descriptor. Empirically, we set τ_s to 0.85.

2.4.2. Pair Distance, Angle, and Direction Verification

There are often many similar features in the actual scene. It is not sufficient to filter out the incorrect matching points by using eigenvalues and descriptions. The distance verification method [20] is a basic verification method. On this basis, we combined the LRF information to further verify the matching pairs. Here, direction and angle filters are proposed to verify the matching pairs. Assuming that (p_i, p'_i), $i = 1, 2, \ldots, n$, are matching feature pairs, p_i is the feature extreme point in current frame coordinate system, p'_i is the query point from the global (scanning model) coordinate system. The angle verification can be expressed as follows:

$$\begin{cases} R = R_{LRF}(p_i) R_{LRF}^T\left(p_j\right)\left(R'_{LRF}\left(p'_i\right) R'^T_{LRF}(p'_j)\right)^T \\ \left(r_x,\ r_y, r_z\right) = \ \text{degree(R)} \\ r_x < \tau_r,\ r_y < \tau_p,\ r_z < \tau_y \end{cases} \tag{9}$$

where the degree() function converts the rotation matrix R to the rotation angle around the axis. In the ideal case, the rotation matrix R is the identity matrix, and $r_x,\ r_y, r_z$ is 0. Further, $\tau_r, \tau_p,\ \tau_y$ indicates the angle threshold of rotation around the axis. The direction verification can be expressed as follows:

$$\begin{cases} \text{dir}\left(p_i, p_j\right) = \ R_{LRF}^T(p_i)\overrightarrow{p_i p_j} \\ \text{dir}\left(p'_i, p'_j\right) = R_{LRF}^T\left(p'_i\right)\overrightarrow{p'_i p'_j} \\ \theta = \ \text{acos}\left(\text{dot}\left(\text{dir}\left(p'_i, p'_j\right), \text{dir}(p_i, p_j)\right)\right) \\ \text{degree}(\theta) < \tau_\theta \end{cases} \tag{10}$$

where $\text{dir}\left(p_i, p_j\right)$ means the direction $\overrightarrow{p_i p_j}$ in the local coordinate system of point p_i, $\overrightarrow{p_i p_j}$ means the direction from point p_i to p_j. Further, $\text{dir}(p'_i, p'_j)$ denotes the direction in the local coordinate system

of point p'_i, and $\overrightarrow{p'_i p'_j}$ represents the direction from point p'_i to p'_j. In the ideal case, the angle between $\mathrm{dir}(p'_i, p'_j)$ and $\mathrm{dir}(p_i, p_j)$ was 0 and cos(0) was 1; therefore, the τ_θ was set to a value close to 1.

The number of consistent pairs for each pair was counted, and the pair for this number was less than the threshold was removed.

2.4.3. Feature Clustering

After the distance, angle, and direction verification, we used the dynamic k-means method for pose clustering to obtain the transformation. The specific details of this algorithm are as follows: Assuming that the relative transformation between a series of matching pairs was $\left\{\mathrm{pose}_{i0}, \mathrm{pose}_{i1}, \ldots, \mathrm{pose}_{j0}\right\}$, we selected pose_{i0} as the first cluster center. Assuming that pose_{i0} was the cluster center of gravity of the first group, we calculated the distance between every remaining pose and the pose cluster group centers. If the distance satisfied the threshold, then the pose was added to the current group and was used to update the center position of the current group. If the distance did not satisfy the threshold, then the pose was viewed as a new group. After all poses were added to the groups, the group whose number was more than the threshold was selected as the target group. If the number of points in the target groups was less than the threshold, the relocation failed.

2.5. Rigid Transformation Calculation

The use of accurate initial parameters can speed up the registration convergence. Therefore, we used the RANSAC method in each clustered group to obtain a better transformation matrix. In each group, the three matching point pairs were randomly sampled, the transformation matrix was calculated by using the three matching point pairs, and the transformation matrix was used to verify whether the each of the rest of the point pairs was an inner point pair. Finally, the transformation matrix with the largest number of interior point pairs was regarded as the correct transformation matrix. Each group was sorted by the number of inner points from large to small. If there was no significant difference in the number of points in the first group and the second group, we concluded that the interference of similar structures existed, and the relocation was considered to have failed. If the numbers of interior points in the first group and the other group were significantly different, the relocation was considered successful. The transformation matrix was refined by using the ICP algorithm with all the interior points.

In the actual scene applications, firstly, the depth map was downsampled, and the feature points were extracted on the downsampled depth map. Similar features were queried from LST, and then the eigenvalue and the descriptor limitations were applied to filter some incorrect matching point pairs. In the verification stage, the distance, angle, and direction verification method was used to filter out the incorrect matching pairs. Then, the cluster method was used to get the groups whose matching points were more than the threshold; finally, the RANSAC method was used to obtain the final result.

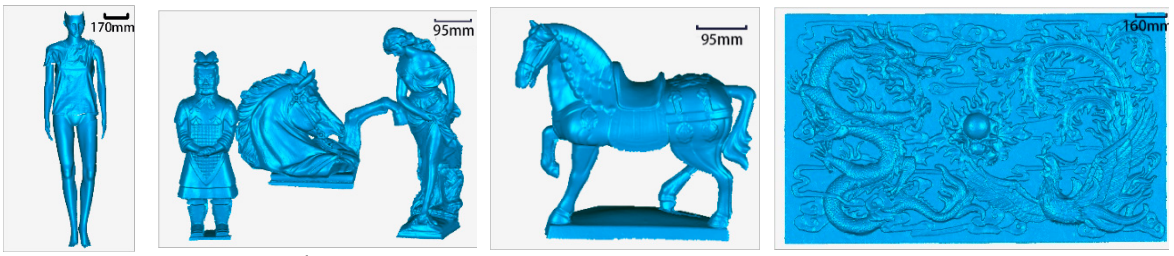
3. Experiments and Results

We evaluated the proposed relocation pipeline on the data collected by a small-range visual scanner. All the experiments were implemented in C++ on a PC equipped with a 2.8-GHz Intel i7 CPU and 16-GB memory. We designed offline and online experiments to verify the real-time and relocation rate of the algorithm.

3.1. Experimental Object and Parameter Set

The four objects were selected from the types of objects that the scanner often scans in actual use. They were used in both the offline experiment and the online experiment, as shown in Figure 2. These four objects had different characteristics. The sizes of "model" and "emboss" were large. The sizes of "small items" and "ceramic horse" were small. The model and ceramic horse objects had smooth

surfaces, and the features were not rich. The surfaces of the "small items" and "emboss" objects were rough, and the features were rich on these surfaces.



(**a**) (**b**) (**c**) (**d**)

Figure 2. Four experiment objects: (**a**) model, (**b**) small items, (**c**) ceramic horse, and (**d**) emboss.

The range of every frame acquired from the scanner was small, the appropriate scanning distance of the scanner used in this study was between 200 mm and 450 mm, and the range was approximately 260 mm × 170 mm at the distance of 400 mm. In the experiments, the resolution of one frame was 1280 × 1024. The downsampling rate s was set to 4, which implied that the size of the depth map that we used was 320 × 256.

On the basis of experience, some of the parameters were set as follows: the radius of the normal calculation was 3 mm. The radius of the LRF calculation was 6 mm, and the width and the height of the window were N = 20 pixels and M = 20 pixels in the selection of the feature extreme point. The value of τ_{area} was set between 0.6 and 0.7. $\tau_{\lambda_{12}}$ and $\tau_{\lambda_{23}}$ were set to 1.5. τ_{λ_2} was set to 0.01. τ_{λ_3} was 0.002. The constraint of the eigenvalues ν_{λ_1} was $\lambda_1/100$, ν_{λ_2} was $\lambda_2/5$, ν_{λ_3} was $\lambda_3/5$, τ_s was 0.85, and τ_r, τ_p, τ_y were all 10°; τ_θ was 30°. The maximum 60 features with the largest λ_3 were selected for each frame.

In the pose clustering, thresholds were set relatively broadly; the thresholds classified into one type were 50 mm and 40°, and the distance threshold and the direction threshold in the feature verification stage were 4 mm and 30°. In the RANSAC phase, the difference in the number of the inner points of the first group and the second group used to determine the similar structures was at least one-third the number of inner points in the first group. The minimum number of correct matching features that determined the success of the relocation was 4.

3.2. Offline Experiment

An offline experiment was designed to demonstrate the real-time and high relocation correct ratio of the proposed relocation pipeline. During data collection, the scanner normally scans the object, tracks the pose and position, and fuses the data. Therefore, each successfully tracked frame has a rotation (R_{global}) and translation matrix (T_{global}). The data of the successfully tracked frames and the rotation and translation matrix of these frames were saved. The saved data constituted an acquisition sequence. The frame data were fed into the relocation pipeline in the order of the acquisition sequence. One frame was selected as the relocation frame for every five frames. Firstly, the features were extracted from the feed frame. If the feed frame was not used for the relocation, the feature points were transformed into global feature points by using the transformation matrix (R_{global}, T_{global}), and these global feature points were inserted into the feature database. If the feed frame was used for the relocation, these feature points were used to query the similar feature points from the feature dataset, and the transformation matrix (R_{reloc}, T_{reloc}) was obtained by using the relocation algorithm. The correctness of the transformation matrix (R_{reloc}, T_{reloc}) was verified according to formula (11). If the transformation matrix is correct, the frame is considered as a correct relocation frame. The ratio of the number of correct relocation frames to the total number of relocation frames is viewed as the relocation correct ratio.

The specific number of acquisition frames for each dataset, the number of frames selected for testing the relocation, the data size, and the total number of feature points are as shown in Table 1.

Table 1. Objects for offline experiments.

Experimental Object	Total Frame	Relocation Frame	Size (mm × mm × mm)
Model	5345	1069	1700 × 477 × 334
Small Items	7320	1464	555 × 761 × 342
Ceramic Horse	2775	555	416 × 474 × 311
Emboss	1746	349	1222 × 2011 × 443

The root mean square (RMS) value was used to determine whether the transformation was correct. The specific formula used was as follows:

$$\begin{cases} \varepsilon_i = (R_{reloc}p_i + T_{reloc} - \left(R_{global}p_i + T_{global}\right) \\ E_{rms} = \sum_{i=0}^{n} \varepsilon_i^2/(n-1) \end{cases} \tag{11}$$

where R_{global} and T_{global} represent the precision rotation and translation parameters, which can transform the current frame into a global one; they were obtained by using the GPU-accelerated ICP algorithm in the data scanning process. R_{reloc} and T_{reloc} represent the rotation matrix and the translation matrix obtained by the relocation algorithm. Further, p_i denotes the feature points extracted from the current frame, n is the number of feature points, and ε_i indicates the distance between the feature points through different transformations. E_{rms} indicates the RMS of the distance. In addition to the RMS statistic, the time cost of the calculation was recorded during the experiment to verify the real-time performance for different numbers of features and frames. In the offline experiment, the time cost of every frame in the relocation, the correct relocation ratio, the relocation RMS error, and the number of feature points were calculated. Figure 3 depicts the relationship between the relocation time and the number of frames. Figure 4 depicts the correct relocation ratio of the four objects. Figure 5 depicts the histogram of the cumulative distribution of the RMS errors. Figure 6 depicts the relationship between the number of feature points and the number of frames.

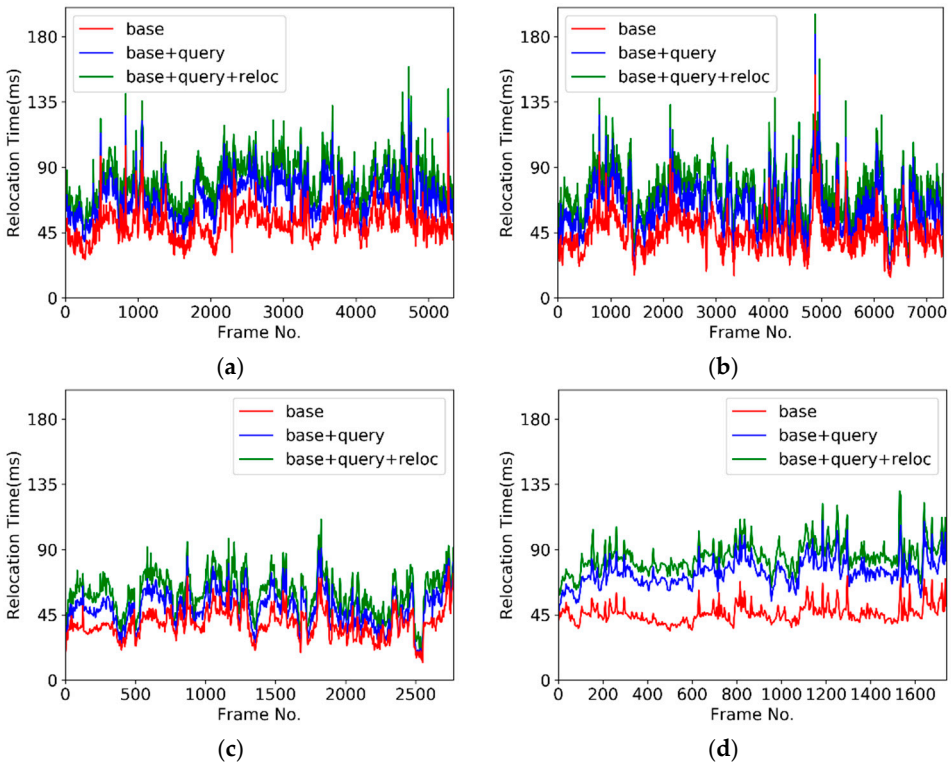


Figure 3. Relocation time cost in the offline experiment: (**a**) relocation time of model, (**b**) relocation of small items, (**c**) relocation time of ceramic horse, and (**d**) relocation time of emboss.

In Figure 3, for the convenience of identification, the time cost of the depth map downsampling and the feature calculation is called the "base" time, and the time cost of the depth map downsampling and the feature calculation and query is called the "base + query" time. As the computational time of the transformation was small, we combined the time of the feature correlation verification and the time of the transformation calculation, and the time cost of the depth map downsampling, feature calculation, query, feature verification, and transformation calculation is called the "base + query + reloc" time.

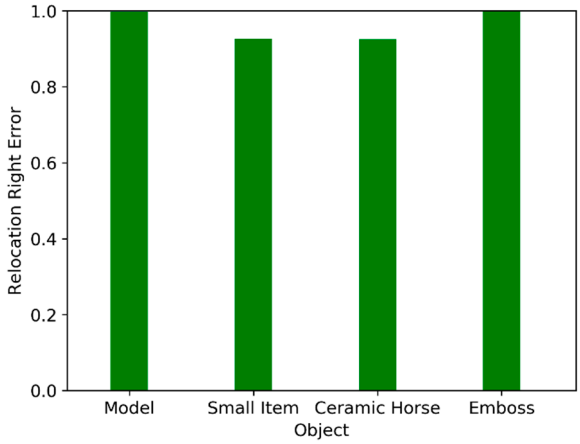


Figure 4. Correct relocation rate of four objects in the offline experiment.

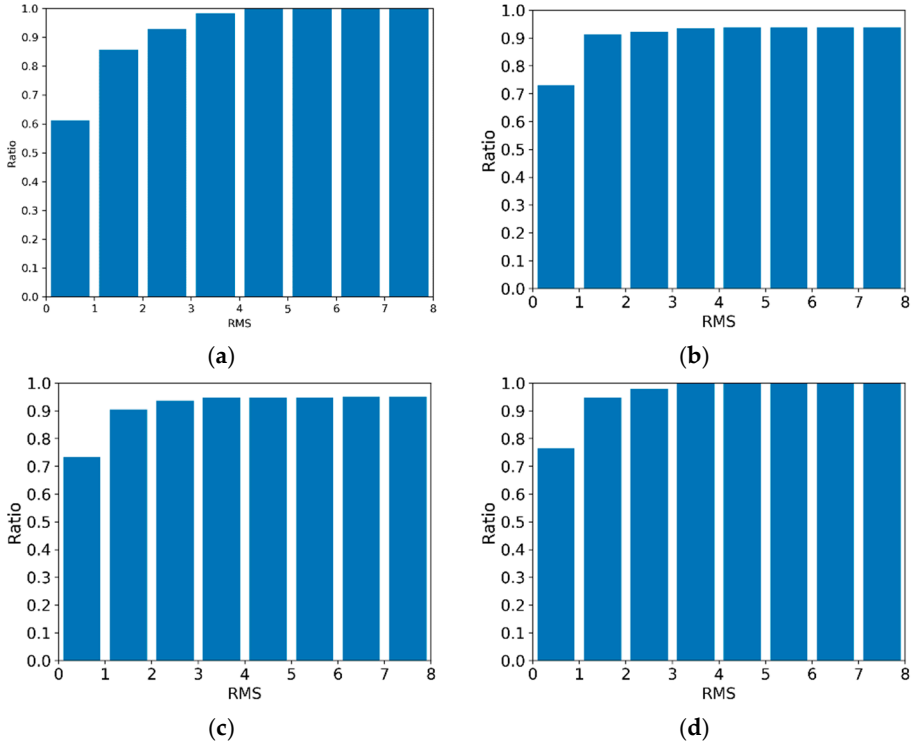


Figure 5. Histogram of cumulative distribution of the relocation RMS errors in the offline experiment. (**a**–**d**) are the RMS error cumulative distribution histogram of the model, small items, ceramic horse, and emboss objects.

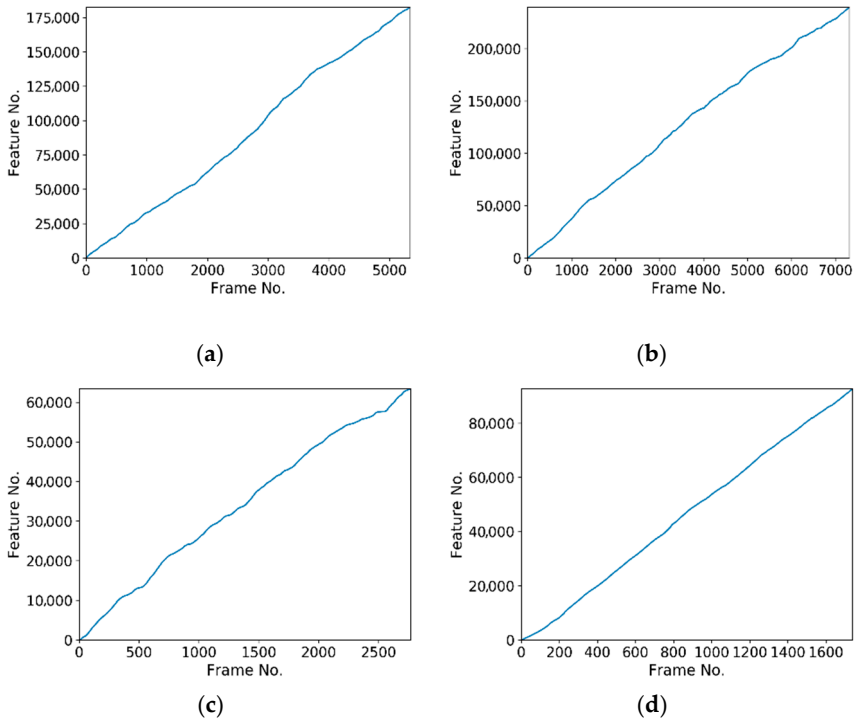


Figure 6. Number of feature points in the offline experiment: (**a**–**d**) are the cumulative number of feature points of the model, small items, ceramic horse, and emboss objects, as the number of frames increases.

3.3. Online Experiment

In the offline experiment, one frame out every five frames was selected as the relocation frame, so each test subject had a large number of relocation frames for testing. This facilitated the statistics of time consumption and relocation rate and was beneficial for observing the trend of time consumption and relocation rate as the numbers of frames and features increased. Typically, relocation algorithms are applied to continue scanning for incomplete data or to re-scan in other places; the regions selected for relocation have randomness. Therefore, we chose some regions for the experiment, which were evenly distributed over the surface of the test object. Firstly, we moved the scanner to scan the entire object and then moved the scanner quickly to the appropriate position that maintained a relatively good distance to the selected test range, and at least two thirds of the selected range fell within the field of view of the sensor. The current frame was displayed in the center of the software view; when the relocation was successful, the coarse transformation matrix was obtained, as described in the previous section. This coarse transformation matrix was used to obtain the model frame data by using the raycasting method; the refine transformation matrix was acquired by aligning the current frame to the model frame, and the current frame data were transformed into global data by using this refine transformation. Thus, we determined whether the global position of the current frame was covered with the selected range through naked eye observation. If it was covered, we considered this relocation to be successful. The ratio of the number of correct relocation locations to the number of all selected locations is viewed as relocation correct ratio.

In the online experiment, we selected the locations evenly to test the performance of the relocation algorithm. As shown in Figure 7, the four objects used for offline testing were used here for the online testing. The selected locations were basically evenly distributed on the surface of the test object.

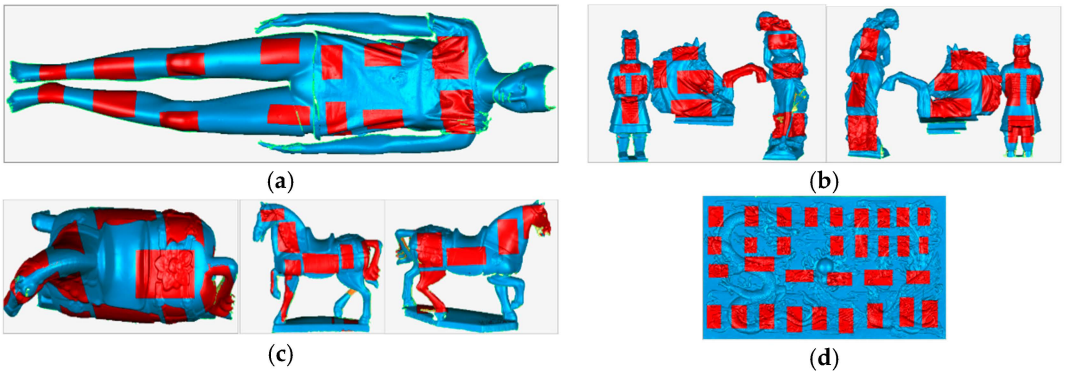


Figure 7. Selected location for the online experiment: the approximate relocation positions are marked by the red rectangles. (**a**–**d**) are selected locations of the model, small items, ceramic horse, and emboss objects.

The information of the experimental data and the experimental results are shown in Table 2. The online experiment was conducted directly after the offline data collection, so the experimental object and the total number of frames were consistent with the offline test. The third column of Table 2 lists the number of selected positions in the four experimental objects. The fourth column lists the statistical average relocation time, and the fifth column lists the statistical relocation rates.

Table 2. Result statistics of online experiments.

Experimental Object	Total Number of Frames	Total Number of Test Locations	Average Relocation Time (ms)	Correct Relocation Rate
Model	5345	14	90	100%
Small Items	7320	28	87	93%
Ceramic Horse	2775	13	76	100%
Emboss	1746	32	82	100%

3.4. Results

In the offline experiment, the relocation time in the scanning process of the four objects is shown in Figure 3. The "base + query + reloc" curve shows the time shift of the relocation with an increase in the number of frames. The relocation time of the model object was kept below 180 ms when the number of frames was 5345. The relocation time of the small items object was kept below 200 ms when the number of frames was 7320. The relocation time of the ceramic horse object was kept below 135 ms when the number of frames was 2775. The relocation time of the emboss object was below 135 ms when the number of frames was 1746. From Figure 3, we inferred that the relocation time did not increase linearly but fluctuated randomly.

The correct relocation ratio of the four objects is shown in Figure 4. We found that the relocation ratios were all greater than 90%. The relocation RMS error cumulative distribution histogram of the four objects is shown in Figure 5. From Figure 5, it is obvious that the trend of the histogram stabilized when the RMS exceeded 5 mm. During the debugging process, we found that it was appropriate to set the threshold of EMS to 5 mm. This ensured that RT was a good initial value. This initial value made the ICP algorithm converge quickly.

From Figure 6, we inferred that the number of feature points of the four objects reached 182,400, 239,515, 63,400, and 92,674, respectively.

In the online experiment, from Table 2, we inferred that the correct relocation ratio of the model, small items, ceramic horse, and emboss objects, respectively, was 100%, 93%, 100%, and 100%. The average relocation times were 90 ms, 87 ms, 76 ms, and 82 ms, respectively.

3.5. Discussion

In the offline experiments, the time consumed by the statistical relocation proved that the proposed relocation algorithm could be used in real time, because each step in the relocation pipeline took less time. Although the number of frames and the number of feature points increased, the relocation time did not increase significantly. This was attributed to the fact that the LST method used a binary tree to organize the feature points. When a feature on the current frame searched for similar features from the database, the feature traversed the LST tree until the leaf node, and on each non-leaf node, the feature traversed the left subtree or right subtree, which determined whether the feature value in the marked dimension was 0. All the features on the leaf node were considered to be similar features. The depth of the tree did not increase very quickly, and the number of judgments was only related to the number of layers, which made the query fast. In the verification step, if there were too many similar features, the verification time was affected, but some simple restrictions, such as limiting the number of similar features, were used to reduce the number of the similar features. Therefore, there was no significant increase in time.

As inferred from Figure 3, we found that both the "base + query" time and the "base + query + reloc" varied with the fluctuations in the base time, and the "query" and "reloc" times were relatively stable. The time of relocation mainly fluctuated with the "base" time. The base time included depth map downsampling, feature calculation, feature extreme point selection, and feature extreme point description. The extracted feature points were generally few, and the time cost of the feature extreme point selection-based grid selection method was small. As the number of feature extreme points per frame was limited to 60, the time cost of the feature extreme point description was extremely small; therefore, the base time was mainly spent on two covariance decompositions (normal calculation and LRF calculation). The reason for this fluctuation was the integrity of the data in the depth map, the occlusion in the data, and other reasons led to the reduction of the invalid data. The less valid the data were, the less the computational time was. The difference in the valid data volume resulted in the fluctuations in the time cost curve.

In the offline test, we observed that the correct relocation rate had reached more than 90%. In theory, if the features on the current frame could search enough correct corresponding features and the verification step removed the wrong matching points, then the correct transformation relationship could be obtained. Further, a correct relocation rate of 90% proved that the constraints of the feature extreme point selection were invalid with respect to ensuring the stability of the feature point position. In addition, one frame out of every five frames was used in the experiment; the ceramic horse and small items objects had more margins in the scanning process than the emboss and model objects (such as the horse legs in the small items object). Therefore, the valid data of the depth map were very small in these tiny places, which led to fewer extracted feature points and led to a relocation failure.

In the online experiment, the relocation time was consistent with the offline test; the reasons for this were analyzed in the offline experiments. The relocation correct ratio was also almost consistent with the offline performance, and the correct relocation rate in the small items object was higher than that in the offline experiment, because the selected regions were not located at the edge or where they could cause occlusions, which made the data relatively complete. These complete data ensured the ability to extract a sufficient number of stable locations. In contrast, the data of one frame at the position of the horse's leg had a large amount of the invalid data in the small items scan, which caused a relocation failure.

There were also some problems in using this relocation pipeline. In the LST trees, the number of feature points stored on each leaf node was limited. If a feature had too many similar features in the database, these similar features were divided into different leaf nodes. Thus, it was difficult to find the correct matching feature from LST, which would result in a relocation failure. In addition, if many different locations were similar to the current frame, relocation using the current frame failed. Therefore, if there were many regions in the scanning object that were similar to the current frame, the current frame might fail to locate the position and the pose of the scanner.

4. Conclusions

We presented a detailed pipeline of a real-time and high-correct-rate relocation algorithm for small-range 3D textureless scanning. The pipeline was divided into four parts, namely feature point calculation, feature database construction and query, corresponding feature verification, and rigid transformation calculation. The experiment proved that the relocation time could be controlled within 200 ms, and the correct relocation rate reached more than 90% in the actual application environment.

The number of feature extreme points and the stability of the feature point position are undoubtedly important factors in determining the correct relocation rate. In this study, the point with the highest curvature in the local region was used as the feature point, and the neighborhood of some of the feature points was disturbed by occlusion or noise. In the future, we intend to use the feature point with the flattest neighborhood to assist with the relocation.

Author Contributions: Conceptualization, S.Z., X.W., F.Z., Y.H., L.G. (Li Gui), and L.G. (Liangxiong Gong); methodology, X.W., F.Z., Y.H., and L.G. (Liangxiong Gong); validation, F.Z.; formal analysis, S.Z., F.Z., and Y.H.; investigation, X.W., F.Z., L.G. (Li Gui), and L.G. (Liangxiong Gong); resources, S.Z.; data curation, X.W., F.Z.; writing—original draft preparation, F.Z.; writing—review and editing, F.Z.; visualization, X.W., F.Z.; supervision, S.Z.; project administration, S.Z.; funding acquisition, S.Z., L.G. (Li Gui).

Funding: This research was supported by National Natural Science Foundation of China (Project Number: 41671452, 41701532), and a China Postdoctoral Science Foundation-funded project (2017M612510).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report with respect to this study.

References

1. GoScan. Available online: https://www.creaform3d.com/en/metrology-solutions/handheld-portable-3d-scanner-goscan-3d (accessed on 4 August 2019).
2. EinScan-Pro+. Available online: http://m.shining3d.com/index.php/scanner/detail/id/18 (accessed on 4 August 2019).
3. GSCAN. Available online: http://www.zg-3d.com/prod_view.aspx?TypeId=71&Id=176&FId=t3:71:3 (accessed on 4 August 2019).
4. Nießner, M.; Zollhöfer, M.; Izadi, S.; Stamminger, M. Real-time 3D reconstruction at scale using voxel hashing. *ACM Trans. Graph.* **2013**, *32*, 1–11. [CrossRef]
5. Shah, S.A.A.; Bennamoun, M.; Boussaid, F. A novel feature representation for automatic 3D object recognition in cluttered scenes. *Neurocomputing* **2016**, *205*, 1–15. [CrossRef]
6. Yang, J.; Xiao, Y.; Cao, Z. Aligning 2.5D Scene Fragments with Distinctive Local Geometric Features and Voting-Based Correspondences. *IEEE Trans Circuits Syst. Video Technol.* **2019**, *29*, 714–729. [CrossRef]
7. Guo, Y.; Sohel, F.; Bennamoun, M.; Wan, J.; Lu, M. An Accurate and Robust Range Image Registration Algorithm for 3D Object Modeling. *IEEE Trans. Multimed.* **2014**, *16*, 1377–1390. [CrossRef]
8. Rock, J.; Gupta, T.; Thorsen, J.; Gwak, J.; Shin, D.; Hoiem, D. Completing 3d object shape from one depth image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 2484–2493.
9. Zhong, Y. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, Kyoto, Japan, 27 September–4 October 2009; pp. 689–696.
10. Besl, P.J.; McKay, N.D. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 239–256. [CrossRef]
11. Serafin, J.; Grisetti, G. NICP: Dense normal based point cloud registration. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 742–749.
12. Myronenko, A.; Song, X. Point set registration: Coherent point drift. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 2262–2275. [CrossRef]
13. Yang, J.; Li, H.; Campbell, D.; Jia, Y. Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *38*, 2241–2254. [CrossRef]

14. Mavridis, P.; Andreadis, A.; Papaioannou, G. Efficient sparse icp. *Comput. Aided Geom. Des.* **2015**, *35*, 16–26. [CrossRef]
15. Petrelli, A.; Di Stefano, L. Pairwise registration by local orientation cues. *Comput. Graph. Forum* **2016**, *35*, 59–72. [CrossRef]
16. Costa, C.M.; Sobreira, H.M.; Sousa, A.J.; Veiga, G.M. Robust 3/6 DoF self-localization system with selective map update for mobile robot platforms. *Robot. Auton. Syst.* **2016**, *76*, 113–140. [CrossRef]
17. Aldoma, A.; Marton, Z.; Tombari, F.; Wohlkinger, W.; Potthast, C.; Zeisl, B.; Vincze, M. Three-dimensional object recognition and 6 DoF pose estimation. *IEEE Robot. Autom. Mag.* **2012**, *19*, 80–91. [CrossRef]
18. Do Monte Lima, J.P.S.; Teichrieb, V. An efficient global point cloud descriptor for object recognition and pose estimation. In Proceedings of the Graphics, Patterns and Images (SIBGRAPI), Sao Paulo, Brazil, 4–7 October 2016; pp. 56–63.
19. Tombari, F.; Salti, S.; Di Stefano, L. Unique signatures of histograms for local surface description. In Proceedings of the European Conference on Computer Vision, Heraklion, Greece, 5–11 September 2010; pp. 356–369.
20. Chen, H.; Bhanu, B. 3D free-form object recognition in range images using local surface patches. *Pattern Recognit. Lett.* **2007**, *28*, 1252–1262. [CrossRef]
21. Buch, A.G.; Yang, Y.; Kr¨uger, N.; Petersen, H.G. In search of inliers: 3d correspondence by local and global voting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 2075–2082.
22. Yulan, G.; Bennamoun, M.; Sohel, F.; Min, L.; Jianwei, W. 3D Object Recognition in Cluttered Scenes with Local Surface Features: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 2270–2287. [CrossRef]
23. Johnson, A.; Hebert, M. Spin-Images: A Representation for 3-D Surface Matching. Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1997.
24. Guo, Y.; Sohel, F.A.; Bennamoun, M.; Lu, M.; Wan, J. TriSI: A Distinctive Local Surface Descriptor for 3D Modeling and Object Recognition. In Proceedings of the GRAPP/IVAPP, Barcelona, Spain, 21–24 February 2013; pp. 86–93.
25. Rusu, R.B.; Blodow, N.; Beetz, M. Fast point feature histograms (FPFH) for 3D registration. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 3212–3217.
26. Tombari, F.; Salti, S.; Di Stefano, L. Unique shape context for 3D data description. In Proceedings of the ACM Workshop on 3D Object Retrieval, Firenze, Italy, 25 October 2010; pp. 57–62.
27. Zhao, B.; Le, X.; Xi, J. A novel SDASS descriptor for fully encoding the information of a 3D local surface. *Inf. Sci.* **2019**, *483*, 363–382. [CrossRef]
28. Novatnack, J.; Nishino, K. Scale-Dependent/Invariant Local 3D Shape Descriptors for Fully Automatic Registration of Multiple Sets of Range Images. In Proceedings of the European Conference on Computer Vision, Berlin/Heidelberg, German, 12–18 October 2008; pp. 440–453.
29. Mian, A.; Bennamoun, M.; Owens, R. On the Repeatability and Quality of Keypoints for Local Feature-based 3D Object Retrieval from Cluttered Scenes. *Int. J. Comput. Vis.* **2010**, *89*, 348–361. [CrossRef]
30. Yang, J.; Zhang, Q.; Xian, K.; Xiao, Y.; Cao, Z. Rotational contour signatures for both real-valued and binary feature representations of 3D local shape. *Comput. Vis. Image Underst.* **2017**, *160*, 133–147. [CrossRef]
31. Chua, C.S.; Jarvis, R. Point Signatures: A New Representation for 3D Object Recognition. *Int. J. Comput. Vis.* **1997**, *25*, 63–85. [CrossRef]
32. Petrelli, A.; Di Stefano, L. On the repeatability of the local reference frame for partial shape matching. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 06–13 November 2011; pp. 2244–2251.
33. Yang, J.; Zhang, Q.; Xiao, Y.; Cao, Z. TOLDI: An effective and robust approach for 3D local shape description. *Pattern Recognit.* **2017**, *65*, 175–187. [CrossRef]
34. Matei, B.; Shan, Y.; Sawhney, H.S.; Tan, Y.; Kumar, R.; Huber, D.; Hebert, M. Rapid object indexing using locality sensitive hashing and joint 3D-signature space estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 1111–1126. [CrossRef]
35. Guo, Y.; Sohel, F.; Bennamoun, M.; Lu, M.; Wan, J. Rotational Projection Statistics for 3D Local Surface Description and Object Recognition. *Int. J. Comput. Vis.* **2013**, *105*, 63–86. [CrossRef]

36. Gálvez-López, D.; Tardos, J.D. Bags of binary words for fast place recognition in image sequences. *IEEE Trans. Robot.* **2012**, *28*, 1188–1197. [CrossRef]
37. Muja, M.; Lowe, D.G. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 2227–2240. [CrossRef]
38. Liu, Q.; Liu, G.; Li, L. Reversed Spectral Hashing. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 2441–2449. [CrossRef]
39. Liu, X.; Fan, X.; Deng, C.; Li, Z.; Su, H.; Tao, D. Multilinear hyperplane hashing. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 5119–5127.
40. Shen, F.; Shen, C.; Liu, W.; Tao Shen, H. Supervised discrete hashing. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 37–45.
41. Muja, M.; Lowe, D.G. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP* **2009**, *2*, 331–340.
42. Papazov, C.; Burschka, D. An efficient ransac for 3d object recognition in noisy and occluded scenes. In Proceedings of the Asian Conference on Computer Vision, Queenstown, New Zealand, 8–12 November 2010; pp. 135–148.
43. Yang, J.; Xiao, Y.; Cao, Z.; Yang, W. Ranking 3D feature correspondences via consistency voting. *Pattern Recognit. Lett.* **2019**, *117*, 1–8. [CrossRef]
44. Yang, J.; Xiao, Y.; Cao, Z. Toward the Repeatability and Robustness of the Local Reference Frame for 3D Shape Matching: An Evaluation. *IEEE Trans. Image Process.* **2018**, *27*, 3766–3781. [CrossRef]